styla extension for hybris commerce

Version: 3.2

Last modified: 29/09/2021

Contents

Overview **3**
- Extension “styla” 3
- Extension “stylaaddon” 3

Quickstart Guide **4**
- Adding the styla extension 4
- Adding the stylaaddon extension 6
- Migration-Guide to version 3.x 8
  - Properties 8
  - StylaSeoService 8
  - MagazineController 8
  - Styla tag 8

The styla extension in detail **9**
- JSON 9
- Dataproviders 9
  - StylaCategoryProvider 11
  - StylaSearchService 11
  - StylaProductProvider 11
- Styla Seo Service 12

The stylaaddon extension in detail **14**
- StylaController (/styla) 14
- StylaMagazineController 14
- Landingpages and StylaLandingPageBeforeViewHandler 15
- CMS 15

Customizations **16**
- I’ve installed the addon extension. How can I test if it is working properly? 17
- We don’t use the hybris Accelerator/Spring MVC for our shop frontend. What do I have to do to get styla running for my shop? 17
- We use a customized data model. What do I have to do to get styla running for my shop? 17
- We don’t use the default Apache SOLR installation of the hybris Accelerator for the product search in our shop frontend. What do I have to do? 18

styla integration for hybris

A full integration of styla with your online shop consists of:

- adding a content page to your online shop where the styla content is rendered
- adding styla's javascript and stylesheet definition to the content page that renders the styla content
- provide endpoints which respond to requests coming from styla in order to expose information about your category structure, productimages in a category and detailed product information for a given product code
- integrate with styla's SEO endpoint, which returns for requested styla content information that is relevant for search engine optimization
- provide a javascript function or endpoint for adding products to the user's cart when called from a magazine page

Overview

The styla integration for hybris consists of two extensions, which simplify the integration of styla with your online shop. The extensions use hybris' Service Layer API which lets you easily implement the interfaces on your own or extend the default implementations provided with this solution.

Extension "styla"

The styla extension is essential. It

- provides classes for convenient conversion of Java objects to JSON and vice versa
- defines the interfaces of service classes which are used by the implementation of the endpoints responding to requests coming from styla
- defines an interface and provides a default implementation for requesting SEO content from styla

Requirements:

The extension relies on hybris' Service Layer Architecture and needs version 5.0 or higher.

Extension "stylaaddon"

The stylaaddon extension is optional and is a sample implementation adding the styla magazine for the hybris B2C Accelerator. It depends on the styla extension, as it implements some of its interfaces. Features of this extension:

- provides a controller implementing the endpoint for exposing data to styla
- provides default implementations of the service interfaces which are used for exposing data to styla
- provides a controller responsible for rendering the magazine page in your online shop
- provides a BeforeViewHandler for rendering Styla's Landingpages in your online shop
- html templates (*jsp, *tag) that render the html for the styla page and add the javascript/stylesheet declarations

- ImpEx files for the creation of CMS items (pages, contentslots, etc.) for the magazine page and landingpage

If your online shop doesn't use the standard hybris accelerator you won't use this extension in your system.

Requirements:

The extension relies on hybris' WCMS system and Acceleratorstorefront and needs version 5.0 or higher.

Quickstart Guide

Disclaimer:

Before performing any step in this manual or deploying this solution in/to your production environment, you should have installed and tested the proper functionality in your non-production. Make sure you have a backup of your working system.

Adding the styla extension

1. Extract the styla directory in styla-hybris-extensions-v3.1.tar.bz2 to {HYBRIS_HOME}/hybris/bin/custom

2. Add to {HYBRIS_HOME}/hybris/config/localextensions.xml this line within the extensions tag:
   …
   <extension name='styla' />
   …

3. Edit {HYBRIS_HOME}/hybris/config/local.properties and override the properties from {HYBRIS_HOME}/hybris/bin/custom/styla/project.properties with the custom values provided to you by styla. Many properties can be specified for site-language combinations of your hybris storefronts, where the property-key follows this syntax: <property-prefixname>.<site uid>.<language ISO code>. For example: styla.username.apparel-de.en would define the username of for the English version of the site apparel-de.

4. Change to directory to {HYBRIS_HOME}/hybris/bin/platform/ and execute

   ant all

As this extension doesn't modify the hybris type system, there is no need to perform a

platform update.

Adding the stylaaddon extension

This is an example how to configure the stylaaddon extension for the hybris vanilla B2C Accelerator.

Adjust the steps accordingly for your individual hybris installation.

1. Extract the stylaaddon directory in styla-hybris-extensions-v3.1.tar.bz2 to {HYBRIS_HOME}/hybris/bin/custom

2. Add to {HYBRIS_HOME}/hybris/config/localextensions.xml this line within the extensions tag:
   …
   <extension name='stylaaddon' />
   …

3. Change to directory to {HYBRIS_HOME}/hybris/bin/platform/ and execute

   ant addoninstall -Daddonnames="stylaaddon"
   -DaddonStorefront.yacceleratorstorefront="yacceleratorstorefront"

   If your storefrontname is different, change "yacceleratorstorefront" accordingly.

4. Edit {yacceleratorstorefront}/web/webroot/WEB-INF/tags/desktop/template/master.tag adding:

   <%@ taglib prefix="styla" tagdir="/WEB-INF/tags/addons/stylaaddon/shared/styla" %>

   to the list of taglib declarations and within the <head> element, add:

   <styla:styla/>

5. Change directory to {HYBRIS_HOME}/hybris/bin/platform/ and execute

   ant all

6. This step is about creating a CMS Page for the Styla Magazine. The addon extension provides the stylaBody.jsp which contains the necessary HTML snippet[1]
   for rendering the magazine.

   The impex located at

   {HYBRIS_HOME}/hybris/bin/custom/stylaaddon/resources/stylaaddon/import/
   contentCatalogs/template/cms-content.impex

[1] The relevant HTML tag is:

generates a two sample CMS pages (a Landingpage and Magazinepage) with the necessary ContentSlot containing the stylaBody.jsp via a JspIncludeComponent. You can use this impex as a starting point, customizing it according to your setup or configure the necessary items in the backoffice. In order to import the adjusted impex, start the hybris platform, open the Hybris Administration Console and go to “Console“ -> “ImpEx Import”.

7. Synchronize your content catalog in order to add the items imported by the cms-content.impex for the staged catalog version to the online catalog version.

As this extension doesn’t modify the hybris type system, there is no need to perform a platform update.

Migration-Guide to version 3.x

If you migrate from a previous version of this plugin to version 3.x you might consider the following changes.

Properties

The properties have been revised. You now only have to define the properties for the user accounts related to a site and the keys for the product api. Please check the chapter “Configuration (project.properties)” for further details.

StylaSeoService

The StyleSeoService now defines the method getSeoForUser which is a generic method to retrieve a SEO object from styla’s SEO Service for a specific styla useraccount. If you use a customized implementation you have to provide an implementation for this method and need to update your classes, that make use of this interface.

MagazineController

The MagazineController has been revised. There is now the abstract AbstractStylaMagazineController class which should be extended by your classes that realize the mapping to a specific path as for instance /magazine. The stylaaddon extension provides an implementation for the mapping of /magazine and /magazin.

Styla tag

The HTML snippet for the styla Javascript should now include the async tag. For this purpose you probably have to update the content of the file {HYBRIS_HOME}/hybris/bin/ext-template/yacceleratorstorefront/web/webroot/WEB-INF/tags/shared//styla.tag accordingly.

The sample code for this tag can be found under {HYBRIS_HOME}/hybris/bin/custom/stylaaddon/acceleratoraddon/web/webroot/WEB-INF/tags/shared/styla/styla.tag.

StylaMagazineContent Component

The HTML snippet rendering the magazine is now located in the stylaBody.jsp file. The cms-content.impex contains a sample setup which generates a JspIncludeComponent (StylaMagazineContentPlaceholder) referencing this jsp.

If you already have a CMS Page for your magazine and want to use the stylaBody.jsp, make sure that you generate the necessary JspIncludeComponent and this component to your CMS Page.

The styla extension in detail

This extension provides elementary functionalities for the integration with styla, which are:

- JSON: classes for convenient conversion of Java objects to JSON and vice versa
- Dataproviders: interfaces of service classes which are responsible for retrieving the right information from hybris commerce suite in order to expose to styla
- Styla Seo Service: interface and default implementation for requesting SEO content from styla
- StylaWebService: a client that communicates with styla's webservice

JSON

For the communication with the styla service JSON is used. The package **com.styla.json** provides classes for marshalling and unmarshalling objects to JSON string and vice versa. For instance, if you request a story from styla, styla will provide information for SEO as JSON, which you can unmarshall by using the Seo class for the creation of the corresponding Seo object. Another example: when you export product search results to styla, you will generate a list of ProductImage objects where a ProductImage contains information such as caption, shop, pageUrl etc. and then provide it as JSON to styla. The ProductImage provides the necessary fields and will be exported to valid JSON.

Dataproviders

Your shop implementation must provide endpoints that respond to requests from styla providing information about your shop ("product api via JSON"). For the current styla API these endpoints are:

- /categories (export category/navigation menu of your shop)
- /products (export products for a given search)
- /product (product detail information)
- /version (provide the version number of the extension)

In order to support customized hybris implementations, this extension defines various interfaces which are used by the classes implementing the endpoints. These interfaces serve as dataproviders whose implementations are responsible for retrieving the required data from the hybris commerce platform.

In the next sections the business logic for information retrieval and the data mapping for each endpoint is briefly described:

The stylaaddon extension implements the endpoints and provides default implementations for these dataprovider interfaces and makes use of the hybris accelerator. In case that a customer doesn't use the hybris accelerator, uses a customized data model or uses a search engine other than hybris' default Apache's SOLR, it might be necessary to implement a custom implementation for these dataprovider interfaces.

StylaCategoryProvider

The StylaCategoryProvider interface is intended to be used for the implementation of the /categories-endpoint. It defines the method getCategories that returns a list of com.styla.json.Category objects. The returned categories must reflect the navigation menu of your shop.

For example, the DefaultStylaCategoryProvider (stylaaddon extension) which implements this interface for the hybris accelerator store, iterates over the hybris CMS components that build up the navigation menu in the shop frontend. For each link that has a (hybris) Category attached, a (styla) Category object is created and filled up with the necessary information. The approach of iterating over the CMS components of the navigation bar is used as the exported categories should represent the stores' navigation menu structure which would not be the case if exporting directly the categories from hybris.

StylaSearchService

The StylaSearchService is intended to be used for the implementation of the /products-endpoint. This interface returns a list of relevant products for given search parameters. The returned search result list contains objects of type com.styla.json.ProductImage.

The search result must correspond to the result for the same search directly executed in the shop, for instance, a text search made in the shop frontend. For this reason, it may be advisable to use the same search implementation as used for the frontend. For example, the DefaultStylaSearchService (stylaaddon extension) uses the ProductSearchFacade of the hybris accelerator.

StylaProductProvider

The StylaProductProvider is intended to be used for the implementation of the /product endpoint. This interface returns detailed product information for a given product code. The returned object is of type com.styla.json.Product.

Styla Seo Service

In order to optimize your styla content pages in your shop for search engines, styla provides an endpoint which returns seo content that can be added to your content page. The interface StylaSeoService defines the method getSeoForUser(String username, String query) which expects the username and query that is sent to the styla service and which returns an object of type com.styla.json.Seo which can be used to add the SEO related content to your page.

The class DefaultStylaSeoService provides an implementation for this interface and essentially forwards requests for a SEO object to the webservice client implementation with its required parameters. The DefaultStylaSeoService uses a cache that stores SEO objects in order to reduce the number of requests to the styla webservice.

In version 2.0 of this extension, the interface has been changed and defines the method getSeoForUser().

StylaWsClient

StylaWsClient represents the remote interface of the style service. DefaultStylaWsClient implements this interface, does the communication with styla's webservice and processes the webservice's responses.

Configuration (project.properties)

In order to configure the styla extension the following properties are relevant and should be overridden in your local.properties file:

Property	**Values/Example**	**Description**
styla.username.<site-uid>.<language-iso> Example: styla.username.apparel-de.de	myshop1	The username for a specific site and language.
styla.productapi.key.<site-uid> Example: styla.productapi.key.apparel-de	KRgmBvgyQOCiC5wd	Authentication key to protect your JSON API endpoint.

For the following properties there should be no need for overriding with custom settings:

Property	**Values/Example**	**Description**
styla.endpoint.seo	http://seo.styla.com/clients	Endpoint of styla's SEO service.

styla.endpoint.version	http://live.styla.com/api/version	Endpoint of styla's version service.
styla.scripts.baseurl	//client-scripts.styla.com/scripts/clients	Base-URL for styla's javascript tag.
styla.styles.baseurl	//client-scripts.styla.com/styles/clients	Base-URL for styla's stylesheet tag.

Configuration (ehcache.xml)

Styla SEO responses are cached in ehcache and is configurable in the file located at *styla/resources/cache/ehcache.xml*

The caching time is configurable changing the timeToIdleSeconds and timeToLiveSeconds attributes.

The stylaaddon extension in detail

If your online shop doesn't use the standard hybris accelerator you won't use this extension. Nevertheless, this documentation might serve as a starting point for your custom implementation.

The stylaaddon extension is optional and is a sample implementation adding the styla magazine to the hybris B2C Accelerator. It depends on the styla extension, as it implements its interfaces and uses the StylaWsClient implementation for requesting SEO content.

The extension contains:

- StylaController: a controller that implements the endpoints for exporting data to styla (/categories, /products, /product, /version)
- StylaMagazineController: a controller that is responsible for rendering the magazine page in your online shop
- Dataproviders: default implementations of the dataprovider interfaces
- html templates (*jsp, *tag) that render the html for the styla page and add the javascript/stylesheet declarations
- ImpEx files for the creation of CMS items (pages, contentslots, etc.) for the magazine page

StylaController (/styla)

This class implements the endpoints for

- /categories
- /products
- /product
- /version

and responds to requests coming from styla. This class makes use of the dataprovider implementations, which are responsible to retrieve the requested information from the hybris commerce platform.

Some endpoints require that styla sends an authentication key as a request parameter ("key"). If this parameter is empty or doesn't match the value defined in your local.properties, a response with HTTP status code 401 (unauthorized) will be returned.

StylaMagazineController

The returned HTML page for the styla magazine must contain the script tag for the styla-javascript and the styletag for the styla-css which are necessary to load the styla magazine on your page and to render the content properly.

Additionally, styla recommends to add additional tags for search engine optimization to the HTML of the magazine page. The content for these tags is requested from styla's webservice. For your

convenience you can use the implementation of the StylaSeoService from the styla extension which returns an object of type com.styla.json.Seo after requesting the information from the webservice.

The AbstractStylaMagazineController prepares the magazine page with all the necessary information. Key functionalities:

- Add the URLs for the magazine's JavaScript and stylesheet to the Model with its correct styla username and version for the respective site.
- Request the SEO information for the requested magazine page from styla's webservice and add the respective head and body content to the page's Model object.

The stylaaddon provides subclasses of this abstract class which only serve the single purpose to respond to a concrete path. For example the StylaMagazineController responds to /magazine (English pages), StylaMagazinController to /magazin (German pages) and StylaPageController responds to /s.

If you want a custom mapping for your magazine page, you should subclass AbstractStylaMagazineController with a RequestMapping for the desired path and configure the page's Model object and determine the view by calling getViewForConfiguredPage.

Landingpages and StylaLandingPageBeforeViewHandler

In order to enable Styla's landingpage feature you have to set the property styla.<sitename>-<language>.enable.landingpage to true (for instance styla.apparel-uk.en.enable.landingpage=true

Property	**Values/Example**	**Description**
styla.<site-uid>.<language-iso>.enable.landingpage Example: styla.apparel-uk.en.enable.landingpage	true	Enables the landingpage feature for a specific site and language.

The StylaLandingPageBeforeViewHandler is a BeforeViewHandler that checks if the current request is hitting the root of the site and if the property to enable landingpages is set. In this case the necessary request to Styla's SeoApi is done and the respective field from the response set to the page model.

CMS

The stylaaddon extension creates a CMS Page and CMS Components for rendering the magazine page. You should configure the Page, Page-Template and ContentSlot if necessary.

One component in the magazine page is CMS component with uid "StylaContentComponent", which contains the HTML snipped for rendering the magazine content.

Configuration (project.properties)

In order to configure the styla extension the following properties are relevant and should be overridden in your local.properties file:

Property	**Values/Example**	**Description**
styla.seo.notfound.page	styla OR hybris	Defines which error page is used when styla's SEO API returns an error.

Customizations

If you don't use the hybris accelerator or a web application framework other than Spring MVC, you have to implement the endpoints that render the styla magazine page (/magazine/*) in your shop and expose your shop data to styla (/styla/*) on your own.

FAQ

I've installed the addon extension. How can I test if it is working properly?

First of all make sure that hybris starts up without error, i.e. pay attention on the logs.

In your browser, check the endpoints (adapt domain and path according to your configuration):

/magazine (example)
https://stylademo-uk.example.com/yacceleratorstorefront/en/magazine/

https://stylademo-uk.example.com/yacceleratorstorefront/en/magazine/story/the-urban-nomad_981034

https://stylademo-uk.example.com/yacceleratorstorefront/en/magazine/user/bob

https://stylademo-uk.example.com/yacceleratorstorefront/en/magazine/tag/fashion

/styla/categories (example)
https://stylademo-uk.example.com/yacceleratorstorefront/styla/categories?key=1234

/styla/products (example)
https://stylademo-uk.example.com/yacceleratorstorefront/styla/products?key=1234

https://stylademo-uk.example.com/yacceleratorstorefront/styla/products?key=1234&search=Jacke

https://stylademo-uk.example.com/yacceleratorstorefront/styla/products?key=1234&category=200000

/styla/product (example)
https://stylademo-uk.example.com/yacceleratorstorefront/styla/product?id=300044624

/styla/version (example)
https://stylademo-uk.example.com/yacceleratorstorefront/styla/version

We don't use the hybris Accelerator/Spring MVC for our shop frontend. What do I have to do to get styla running for my shop?

You basically have to add the styla extension to your hybris configuration and implement the dataprovider interfaces, the endpoint for the stylamagazine and the RESTful webservices on your own. Please check chapter The styla extension in detail and The stylaaddon extension in detail for more details.

Relevant interfaces: StylaProductProvider, StylaSearchService, StylaCategoryProvider

We use a customized data model. What do I have to do to get styla running for my shop?

You basically have to write your custom implementation of the dataprovider interfaces.

Relevant interfaces: StylaProductProvider, StylaSearchService

We don’t use the default Apache SOLR installation of the hybris Accelerator for the product search in our shop frontend. What do I have to do?

You basically have to write your custom implementation of the dataprovider interfaces.

Relevant interfaces: StylaProductProvider, StylaSearchService